A PARALLEL PARTICLE-IN-CELL CODE FOR EFFICIENTLY MODELING PLASMA WAKEFIELD ACCELERATION: QUICKPIC

Chengkun Huang , Viktor Decyk, Shuoqin Wang, Evan S. Dodd, Chuang Ren, Warren B. Mori
University of California Los Angeles, Los Angeles, CA 90095
Tom Katsouleas
University of Southern California, Los Angeles, CA 90089
James Cooley, Tom Antonsen Jr.
University of Maryland, College Park, MD 20742

Abstract:

There has been much renewed interest in plasma wakefield acceleration. This is due in part to the possibility of using it as an energy doubler, i.e., an "afterburner", stage at the end of an existing linear collider such as SLC[1]. The generation of plasma wave wakes due to intense particle beams is highly nonlinear and therefore particle models are required to study it. Unfortunately, even on the largest supercomputers, it is still not practical to use standard particle-in-cell (PIC) codes such as OSIRIS[2] to model an entire 100 GeV energy doubler stage. Luckily, there is a disparate difference in time scales between how the drive beam evolves and the period of the wakefield oscillation. We have recently developed a three-dimensional, parallelized PIC code called QuickPIC which makes a frozen field approximation. This code embeds a two-dimensional electrostatic parallel PIC code inside of a three-dimensional electrostatic parallel PIC code. In this paper, we will review how a standard electromagnetic PIC code such as OSIRIS works, describe the frozen field approximation and then describe how QuickPIC works. Preliminary results will be presented, which include benchmarking QuickPIC results against OSIRIS results.

I. INTRODUCTION

The challenge of modeling any plasma physics problem is to self-consistently evolve the electromagnetic fields through Maxwell's equations and the trajectories and currents (and charge densities) of the particles through the relativistic equations of motion. The technique which uses the least physics approximations is the fully explicit particle-in-cell (PIC) technique in which the full set of Maxwell's equations is solved together with the individual trajectories of each plasma particle. While PIC codes make the fewest approximations of any plasma model, they are also the most

huangck@ee.ucla.edu

computationally demanding. Therefore, to use such codes on even some simple problems requires that they be implemented using parallelized algorithms. At UCLA we have been developing parallel PIC algorithms since 1987 and we currently have a variety of fully parallelized PIC codes. In some of these codes Maxwell's equations are solved in Fourier space while in others they are solved in real space using finite difference equations. These also include electrostatic and fully electromagnetic codes. One of our electromagnetic codes called OSIRIS has been used extensively to model advanced accelerator concepts[3,4,5,6,7,8,9].

To model the excitation of plasma wave wakes due to an intense particle beam traversing a plasma requires the use of PIC codes since particles trajectories are nonlinear and cross through each other. The acceleration gradient in these wakes may approach 100 GeV/m. It turns out that to model a beam propagating through a few meters of 10^{16}cm^{-3} plasma, i.e., a 100 GeV stage, using our standard code OSIRIS will take ~100,000 hours of CPU time. Therefore, from a practical standpoint we need to develop a different type of code to model this problem.

Fortunately, the beam evolves slowly while it moves through the meter of plasma. We can therefore develop an approximate code which exploits this, while still keeping a high degree of accuracy. We have started this development by using pieces of existing parallel PIC codes which exist at UCLA. This was possible due to a new frameworks strategy for parallel algorithms which we are developing.

In this paper, we will first describe the equations to be solved and the frozen field approximation. We will then briefly describe the frameworks concept and how it was used to construct QuickPIC. Last we will give some preliminary results and some comparisons between OSIRIS and QuickPIC.

II. QUASI-STATIC APPROXIMATION

QuickPIC is based on the quasi-static approximation. We start from the Maxwell equations in Lorenz gauge,

$$(\frac{1}{c^2}\frac{\partial^2}{\partial t^2} - \ ^2)\phi = 4\pi\rho \qquad (1)$$

$$(\frac{1}{c^2}\frac{\partial^2}{\partial t^2} - \ ^2)\mathbf{A} = \frac{4\pi}{c}\mathbf{j} \qquad (2)$$

In (x, y, s, ξ) coordinates, where $s = z$ (z is the direction in which the beam is moving), $\xi = t - z/c$, quasi-static approximation can be taken, i.e., $\partial_s \approx 0$. Then a set of full quasi-static equations can be written as,

$$^2\phi = -4\pi\rho, \qquad (3)$$

$$^2\mathbf{A} = -4\pi\mathbf{j}/c, \qquad (4)$$

$$\mathbf{A} = -\frac{\partial \Psi}{\partial \xi}, \tag{5}$$

with the equations of motion being:

$$\frac{d\mathbf{P}_b}{ds} = q_b \quad \Psi, \tag{6}$$

$$\frac{d\mathbf{P}_e}{d\xi} = \frac{q_e}{1 - V_{e//}/c}[\mathbf{E} + (\mathbf{V}_e \times \mathbf{B})], \tag{7}$$

In Eq. (5) and (6), $\Psi = \phi - A_{//}$, where $A_{//}$ is the longitudinal component of vector potential. q is the charge of particle and ρ is the charge density. **V**, **P** are velocity and momentum and the subscript b, e denote beam electron and plasma electron respectively.

Notice that the longitudinal beam current dominates the transverse currents of the beam and the plasma. To the lowest order, one can neglect the transverse current $\mathbf{j}$, and hence the transverse component of the vector potential $\mathbf{A}$. The longitudinal current by the plasma can also be neglected to the lowest order. The above quasi-static equations can be reduced to:

$$^2\phi = -4\pi\rho = -4\pi(\rho_b + \rho_e + \rho_{ion}), \tag{8}$$

$$^2\Psi = -4\pi(\rho_e + \rho_{ion}), \tag{9}$$

$$\frac{d\mathbf{P}_b}{ds} = q_b \quad \Psi, \tag{10}$$

$$\frac{d\mathbf{V}_e}{d\xi} = q_e \quad \phi \cdot \tag{11}$$

The equations above only involve two dimensions, which are perpendicular to the beam propagation direction, so Eqs. (8) and (9) can be solved in 2D space. Once the potentials are calculated, the velocity and position of particles can be updated using Eqs. (10) and (11). For plasma electrons, this is done for every time step $\Delta\xi$, which needs to resolve the plasma frequency. While for beam electrons, the time step is Δs, which only needs to resolve the betatron frequency or the hosing growth.

The above lowest order algorithm is identical to that of D. H. Whittum [10] and is adopted in the first phase of QuickPIC development. Eqs. (8) and (9) are solved in Fourier space by a parallel Poisson solver, either with periodic or conducting boundary conditions. We have also added three major corrections. These corrections include: (a) a relativistic plasma electron pusher; (b) a correction to the plasma charge density due to the longitudinal motion of plasma electrons by using the following deposition scheme:

$$\rho_e = \frac{1}{V}\sum_i \frac{q_{ei}}{1 - V_{e//i}/c}, \tag{12}$$

and (c) an addition of the parallel current of plasma,

$$\mathbf{j}_{e//} = \frac{1}{V}\sum_i \frac{q_{ei}V_{e//i}}{1 - V_{e//i}/c}\hat{z}. \tag{13}$$

These three corrections can be explicitly turned on or off from the input.

Currently, QuickPIC can achieve results reasonably similar to those from more computational intensive full electromagnetic code such as OSIRIS for medium beam/plasma density ratio. We find that in this parameter regime QuickPIC gives results that are within 10s of percent of those from OSIRIS but at a saving of 1000 times in computing time. As the beam density goes higher, transverse current starts to play an important role, and accuracy of the reduced quasi-static description deteriorates.

The second phase development of QuickPIC is still ongoing, and the goal is to implement the full quasi-static equations as described in Eqs. (3)-(7) as well as the ability to model laser drivers instead of particle beam drivers. An efficient algorithm for properly time-centering the full quasi-static equations may be essential to the development. Also, a new framework for object-oriented programming is being introduced at this phase, which we will describe in the next section.

III. FRAMEWORKS

Frameworks are an emerging technology for code reuse in development of complex software systems [11,12]. They are generally based on an object-oriented design, with interacting classes that can be specialized to produce custom applications. They have been largely used in developing business software, with only a few examples in scientific computing [13]. While frameworks are commonly implemented in an object-oriented language, this is not a requirement.

Although there are many definitions of a framework, the working definition we have adopted is that a framework is a unified environment containing all the components needed for writing code for a specific problem domain. Its goal is the rapid construction of new codes by reusing trusted modules. Although our framework can be used to write a large “mega-code,” it is designed to provide “Lego” pieces for rapid construction of new codes. Our framework will support multiple numerical methods, different physics approximations, different numerical optimizations and implementations for different hardware. Above all, it is designed to hide the complexity of parallel processing.

Our approach is to write powerful high level classes that can easily be reused for those parts of the code which new programmers such as graduate students do not intend to modify. These high level classes provide simple interfaces that encapsulate the implementation details of a large block of code. Invariably, students need to modify some high level class for their research needs. To simplify this process, we find that it is useful to build high-level classes from middle-layer helper classes. These helper classes contain descriptors of data, but not the data themselves.

We have begun implementation of such a framework for parallel plasma Particle-in-Cell (PIC) simulations. The current framework is designed to support two and three dimensional parallel codes. It uses trusted legacy Fortran77 subroutines as the lowest layer. These legacy subroutines are well debugged and provide most of the basic functionality, but are not intended to be modified greatly. The most CPU time-consuming parts of a PIC code are the particle push and charge deposit. These subroutines have been very carefully written to provide the highest performance possible[14]. Figure 1

illustrates the kind of domain decomposition supported by the framework to give optimum load balancing on parallel processors.

For the first phase of our code development, we used existing 2D code and embedded it into a 3D one. The 2D Fortran 90 routine calls the wrappers, which actually pass parameters to legacy Fortran 77 subroutines, to solve Eqs. (8) and (9) and push the plasma electrons. The calculated forces are returned to the 3D main program where the beam electrons are updated. In the next phase, these legacy Fortran 77 subroutines will be encapsulated into classes which will make the design of 2D and 3D routines more simplified.

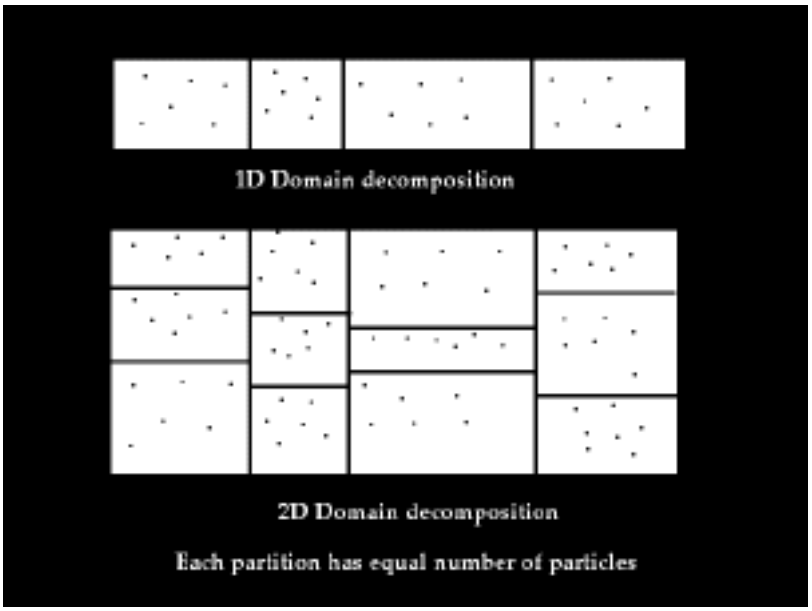

Figure 1: Diagram shows how space can be partitioned among different processors in a parallel PIC code.

III. PRELIMINARY RESULTS

We have use QuickPIC to conduct some benchmarking runs based on E157 / afterburner beam and plasma parameters. For the simulations of E157 experiment, plasma density was 2.1×10^{14} cm^{-3}. The beam charge was 1.8×10^{10} electrons and the charge density had a Gaussian distribution with σ_r=0.07mm and σ_ξ=0.63mm. The energy of beam was 30Gev (γ=60,000). The dimensions of the simulations were 8×8×17.6 c/ω_p (2.9×2.9×6.4mm), divided into 256×256×256 cells. 128×128×320 particles were used for the beam and 512×512 particles for the plasma. For the afterburner runs, plasma density was 2.1×10^{16} cm^{-3}, spot size σ_r=0.007mm and σ_ξ=0.045mm (the beam density normalized to the plasma density is about ten times higher.) The dimensions were 16×16×16 c/ω_p (0.59×0.59×0.59mm), with 256×256×256 cells. Other parameters were the same as E157 simulations. Figure 2 shows comparison of the accelerating fields from QuickPIC and OSIRIS for the E157 benchmarking run and Figure 3 shows the same comparison for Afterburner benchmarking run.

QuickPIC has been used to study the electron hosing instability. Figure 4 shows the growth of centroid oscillation from a QuickPIC Afterburner run.

IV CONCLUSION

The algorithm and structure of a new parallelized quasi-static code called QuickPIC was described. We have presented the preliminary results from benchmarking and hosing runs. The full set of quasi-static equations and new framework will be integrated into QuickPIC and we hope it will then reproduce the results of fully electromagnetic code.

This work was supported by DOE under contract number DE-FC02-01ER41179 and by NSF grants Phy-0078508 and ECS-9617089, and by LLNL under grant # W-07405-ENG48.

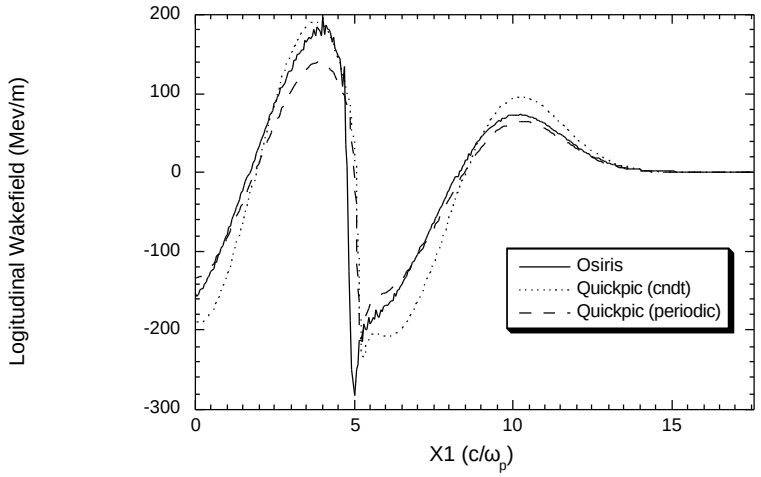

Figure 2: Longitudinal accelerating field (E157 parameters) from QuickPIC(dot-dash, conducting boundary condition; dash, periodic boundary condition) and OSIRIS (solid)

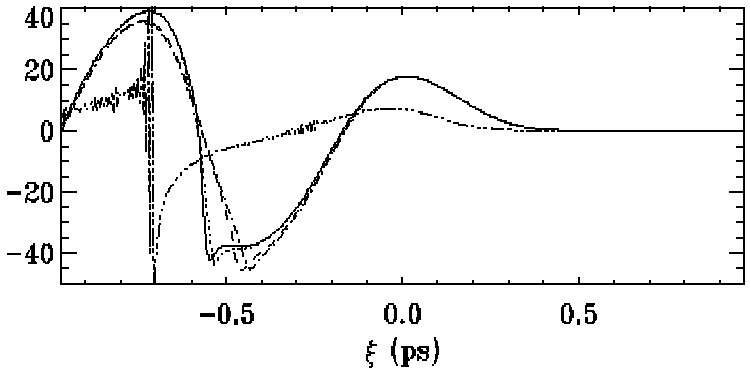

Figure 3: Longitudinal accelerating field (Afterburner parameters) from QuickPIC(solid and dash, ploting accelerating field at different time steps) and OSIRIS (dot-dash), the Y axis is accelerating field in Gev/m.

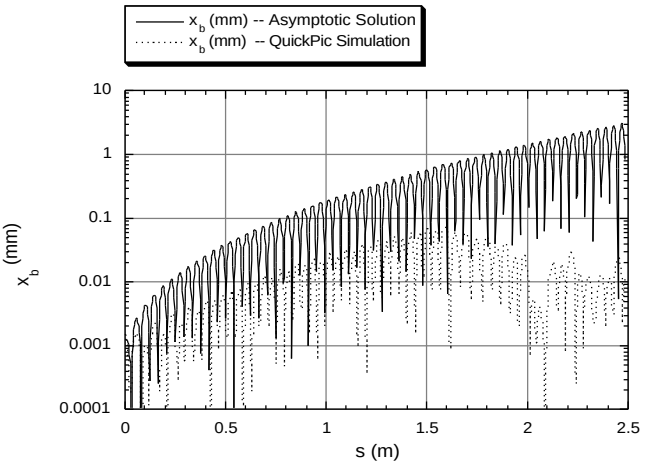

Figure 4: beam centroid oscillation (Afterburner parameters)

REFERENCE

[1] S. Lee et. al., Phys. Rev. STAB, 5, 011001,(2002)

[2] R. G. Hemker et. al., Proc. 1999 Part. Accel. Conf., (1999)

[3] S. Lee et. al., Phys. Rev. E, 6, 7014, (2000)

[4] R. G. Hemker et. al., Phys. Rev. STAB, 3, 061301, (2000)

[5] C. Ren et. al., Phys. Rev. E, vol. 63, pp.026411/1-8, (2001)

[6] P. Muggli et. al., Nature, 411, 32 (2001)

[7] C. Ren et. al., Phys. Rev. E, vol. 64, 067401 (2001)

[8] S. Lee et. al., Phys. Rev. E, vol. 64, 045501,(2001)

[9] E. S. Dodd et. al., to be appear in Phys. Rev. Lett.

[10] D. H. Whittum, Phys. Plasmas, 4, 1154 (1997)

[11] E. Gamma et. al., Design Patterns, Addison-Wesley, Boston. 1995.

[12] M. E. Fayad et. al., Building Application Frameworks, John Wiley & Sons, New York, 1999.

[13] Common Component Architecture Forum, http://www.cca-forum.org/.

[14] V. K. Decyk et. al., Computers in Physics 10, 290 (1996).